Java-Creating a new file

Project Abstract

The purpose of this project is to demonstrate how to create files in Java using the File class. This exercise helps in understanding the process of checking file existence, creating new files, and handling exceptions related to file operations. Java provides an API for handling file creation and manipulation, and this project aims to guide students through the file creation process in a practical way.

This project focuses on:
1. Understanding how to create files in Java using the File class.
2. Implementing file existence checks before creating a new file.
3. Handling exceptions that may occur during file operations using try-catch blocks.
4. Demonstrating the usage of the createNewFile() method to create a file in the local file system.

Tasks Overview

Task 1: Create a File Using the File Class

Objective: Objective: Create a file using Java's File class and check whether the file already exists before creating it.

Detailed Description: In this task, students will use the File class to create a new file with a specific name. The file's name should be "newFile.txt". Before attempting to create the file, the program will check if a file with that name already exists in the directory. If the file does not exist, it will create the file. If the file already exists, it will print a message indicating so.

Steps:
1. **Create a New File:**
   - Declare a File object named `file` using the constructor `new File("newFile.txt")`. This will represent the file "newFile.txt", which you will attempt to create.

2. **Check if the File Exists:**
   - Use the `createNewFile()` method to attempt to create the file. This method will return `true` if the file was successfully created, and `false` if the file already exists in the working directory.

3. **Handle the Outcome:**
   - If the file is created, print the message "File created: newFile.txt". This message confirms that the file did not exist before and was created successfully.
   - If the file already exists, print the message "File already exists.". This message indicates that no new file was created since the file already exists in the directory.

4. **Handle Errors:**
   - If an exception occurs during file creation, such as a lack of permissions or insufficient disk space, the program will enter the `catch` block. You should catch the `IOException` and print the message "An error occurred while creating the file.". Additionally, print the exception stack trace using `e.printStackTrace()` to provide detailed information about the error.

Execution Steps to Follow:

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) Terminal New Terminal.**
3. **This editor Auto Saves the code.**
4. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**
5. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
6. **To run your project use command:**
   mvn compile exec:java -Dexec.mainClass="com.yaksha.assignment.FileCreation"
7. **To test your project test cases, use the command**
   mvn test

8. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**